Package ‘gretel’

October 13, 2022

Title  Generalized Path Analysis for Social Networks
Version  0.0.1
Date  2019-08-09

Description  The social network literature features numerous methods for assigning value to paths as a function of their ties. 'gretel' systemizes these approaches, casting them as instances of a generalized path value function indexed by a penalty parameter. The package also calculates probabilistic path value and identifies optimal paths in either value framework. Finally, proximity matrices can be generated in these frameworks that capture high-order connections overlooked in primitive adjacency sociomatrices. Novel methods are described in Buch (2019) <https://davidbuch.github.io/analyzing-networks-with-gretel.html>. More traditional methods are also implemented, as described in Yang, Knoke (2001) <doi:10.1016/S0378-8733(01)00043-0>.

Maintainer  David Buch <davidbuch42@gmail.com>

URL  https://github.com/davidbuch/gretel

BugReports  https://github.com/davidbuch/gretel/issues

License  GPL-3

Depends  R (>= 3.0)

Imports  Rcpp (>= 1.0.0), ResistorArray (>= 1.0-32)

LinkingTo  Rcpp

Suggests  knitr, rmarkdown, testthat (>= 2.1.0)

VignetteBuilder  knitr

LazyData  true

RoxygenNote  6.1.1

NeedsCompilation  yes

Author  David Buch [aut, cre] (<https://orcid.org/0000-0002-4574-0075>)

Repository  CRAN

Date/Publication  2019-08-22 12:00:02 UTC
all_opt_gpv

**Description**

Identify the path of optimal generalized path value from every source to every target in sociomatrix.

**Usage**

```r
all_opt_gpv(sociomatrix, p = Inf, node_costs = NULL)
```

**Arguments**

- `sociomatrix`: a nonnegative, real valued sociomatrix.
- `p`: a nonnegative real number that sets the 'p-norm' parameter for generalized path value calculation.
- `node_costs`: a list of costs, in order, of all nodes represented in the sociomatrix, all are assumed 0 if unspecified

**Value**

All optimal paths from source to target nodes in sociomatrix. To minimize memory usage, paths are returned as a list of trees in Dijkstra's format. Specific paths can be unpacked with unpack as described in the example below.
all_opt_ppv

See Also

gpv to calculate the value of a user-specified path, opt_gpv to identify the optimal path from a single source node to a single target node

Description

Identify the path of optimal probabilistic path value from every source to every target in sociomatrix.

Usage

\[
\text{all_opt_ppv}(\text{sociomatrix}, \text{odds_scale} = 1, \text{odds_scale_by_node} = \text{NULL})
\]

Arguments

- **sociomatrix**: a nonnegative, real valued sociomatrix.
- **odds_scale**: a nonnegative real number indicating the observed tie strength value that corresponds to 1-1 transmission odds
- **odds_scale_by_node**: sets a transfer odds scale for each node in a probabilistic path value calculation.

Value

All optimal paths from source to target nodes in sociomatrix. To minimize memory usage, paths are returned as a list of trees in Dijkstra’s format. Specific paths can be unpacked with unpack as described in the example below.

See Also

ppv to calculate the value of a user-specified path, opt_ppv to identify the optimal path from a single source node to a single target node

Description

Calculates the binary distance of a user-specified network path through a network, if all edges exist. Otherwise, returns \( \text{Inf} \) to signify infinite distance.

Usage

\[
\text{binary_distance}(\text{sociomatrix}, \text{path})
\]
Arguments

- **sociomatrix**: a nonnegative, real valued sociomatrix.
- **path**: an integer vector of node indices from `sociomatrix`.

Examples

```r
## Calculate binary distance along a path in a sociomatrix
binary_distance(YangKnoke01, path = c(1,2,5))

## This path doesn't exist
binary_distance(YangKnoke01, path = c(1,2,4,5))
```

---

**BuchDarrah19**

*Example data for gretel*

Description

A sociomatrix encoding tie strengths among five nodes

Usage

`BuchDarrah19`

Format

A numeric matrix with 5 rows and 5 columns

Source

<DOI:10.1016/j.socnet.2010.03.006>

---

**dijkstra_inf**

*Find the shortest L-Inf norm paths to other vertices*

Description

Find the shortest L-Inf norm paths to other vertices

Usage

`dijkstra_inf(dist, src)`

Arguments

- **dist**: A matrix of distances between nodes
- **src**: An integer vertex ID
**dijkstra_nodes**

*Value*

A numeric vector, entry \( i \) of which is the vertex immediately preceeding vertex \( i \) in the shortest path leading to \( i \). Full paths must be constructed recursively.

---

**flament_average_path_length**

*Yang and Knoke's Average Path Length*

*Description*

Calculates 'APL' (Average Path Length) as defined in Yang, Knoke (2001). Called `flament_average_path_length` in homage to A.C. Flament, who defined path length in 1963.

*Usage*

`flament_average_path_length(sociomatrix, path)`

*Arguments*

- `sociomatrix`: a nonnegative, real valued sociomatrix.
- `path`: an integer vector of node indices from `sociomatrix`. 

---

**dijkstra_nodes**

*Find the shortest paths to other vertices*

*Description*

Find the shortest paths to other vertices

*Usage*

`dijkstra_nodes(dist, src, node_costs)`

*Arguments*

- `dist`: A matrix of distances between nodes
- `src`: An integer vertex ID
- `node_costs`: a list of costs, in order, of all nodes represented in the sociomatrix, all are assumed 0 if unspecified

*Value*

A numeric vector, entry \( i \) of which is the vertex immediately preceeding vertex \( i \) in the shortest path leading to \( i \). Full paths must be constructed recursively.
See Also

flament_path_length

Examples

## Calculate 'APL' of a path in a sociomatrix
flament_average_path_length(YangKnoke01, path = c(1,2,5))

## This path doesn't exist
flament_average_path_length(YangKnoke01, path = c(1,2,4,5))

flament_path_length
Flament's Path Length Measure

Description

Calculates path length as defined in Flament (1963). That is, sums the values of each edge in the path, if all edges exist. Otherwise, returns NA.

Usage

flament_path_length(sociomatrix, path)

Arguments

sociomatrix a nonnegative, real valued sociomatrix.
path an integer vector of node indices from sociomatrix.

See Also

flament_average_path_length

Examples

## Calculate Flament's Path Length along a path in a sociomatrix
flament_path_length(YangKnoke01, path = c(1,2,5))

## This path doesn't exist
flament_path_length(YangKnoke01, path = c(1,2,4,5))
generate_proximities

Generate a Proximity Matrix

Description

Generates a proximity matrix in one of three modes:

'ogpv' Optimal Generalized Path Value. Entry $i,j$ of the proximity matrix will equal the optimal 'gpv' among all paths connecting node $i$ to node $j$.

'oppv' Optimal Probabilistic Path Value. Entry $i,j$ of the proximity matrix will equal the optimal 'ppv' among all paths connecting node $i$ to node $j$.

'sconductivity' Social Conductivity (Random Walk Probability). If each tie strength recorded in sociomatrix is taken to be analogous to the conductivity of an electrical component, $i,j$ of the proximity matrix will equal total conductivity of all paths from node $i$ to node $j$.

Usage

generate_proximities(sociomatrix, mode = c("ogpv", "oppv", "sconductivity"), p = Inf, node_costs = NULL, odds_scale = 1, odds_scale_by_node = NULL)

Arguments

sociomatrix a nonnegative, real valued sociomatrix.
mode a selection of 'ogpv', 'oppv', or 'sconductivity'
p if mode is 'ogpv', determines 'p-norm' parameter for generalized path value calculation.
node_costs if mode is 'ogpv', assigns transmission costs to vertices within the sociomatrix.
odds_scale if mode is 'oppv', sets a global transfer odds scale for probabilistic path value calculation.
odds_scale_by_node if mode is 'oppv', sets a transfer odds scale for each node in a probabilistic path value calculation.

See Also

gpv, ppv

Examples

## Generate a proximity matrix in each mode
## Optimal Generalized Path Value
generate_proximities(YangKnoke01, mode = "ogpv", p = Inf, node_costs = c(1,3,3,2,1))

## Optimal Probabilistic Path Value
generate_proximities(YangKnoke01, mode = "oppv", odds_scale = 2)
```r
## Sconductivity
generate_proximities(YangKnoke01, mode = "sconductivity")
```

---

**Description**

Calculates the generalized path value of a user-specified path through sociomatrix. Parameter \( p \) sets the \( p \)-norm used in calculation.

**Usage**

```
gpv(sociomatrix, path, p = Inf, node_costs = NULL)
```

**Arguments**

- `sociomatrix`: a nonnegative, real valued sociomatrix.
- `path`: an integer vector of node indices from sociomatrix.
- `p`: a nonnegative real number that sets the \( p \)-norm parameter for generalized path value calculation.
- `node_costs`: a list of costs, in order, of all nodes represented in the sociomatrix, all are assumed 0 if unspecified

**Details**

As a rule of thumb, \( p \) close to 0 will downweight the impact of particular tie strengths and upweight the impact of binary path length. \( p \) equal to infinity will recapitulate the traditional path value measure of Peay (1980) and is therefore the default. In other words, the value of a path under \( p = \infty \) will be the value of the weakest tie. The value of the same path under \( p = 0 \) will be the inverse of its binary length.

**See Also**

`opt_gpv` to identify the path of optimal \( 'gpv' \) between two nodes and `all_opt_gpv` to identify the optimal paths between all pairs of nodes. Calling `generate_proximities` with \( mode = 'gpv' \) returns a matrix \( 'gpv' \) values for the optimal paths between all pairs of nodes.

**Examples**

```r
## Calculate gpv along a path in a sociomatrix
gpv(YangKnoke01, path = c(1,2,5), p = 1)
```

```r
## The same calculation, with nonzero node costs
gpv(YangKnoke01, path = c(1,2,5), p = 1, node_costs = c(1,3,3,2,1))
```
## This path doesn't exist

gpv(YangKnoke01, path = c(1,2,4,5), p = 0)

---

### Description

This package contains two categories of functions. The first category is concerned with assigning values to user specified paths, while the second identifies paths of optimal value.

### Details

Key functions in the path value calculation category are - `gpv`, which calculates Generalized Path Value - `ppv`, which calculates Probabilistic Path Value - `binary_distance`, `peay_path_value`, `flament_path_length`, `peay_average_path_value` and `flament_average_path_length`, which calculate path value measures described in Yang, Knoke (2001). - `generate_proximities`, which generates a matrix of values representing the measures of optimal paths from each source node (row index) to each target node (column index).

Key functions in the optimal path identification category are - `opt_gpv`, which identifies the path of optimal Generalized Path Value from a particular source node to a particular target node - `opt_ppv`, which identifies the path of optimal Probabilistic Path Value from a particular source node to a particular target node - `all_opt_gpv`, which identifies the 'gpv'-optimal paths from every source node to every target node - `all_opt_ppv`, which identifies the 'ppv'-optimal paths from every source node to every target node - `unpack`, which unpacks the Dijkstra-format encoded shortest paths returned by `all_opt_gpv` and `all_opt_ppv`. See their help pages for details.

---

### OpsahlEtAl10

**Example data from Opsahl, Agneessens, Skvoretz (2010)**

### Description

A sociomatrix encoding tie strengths among five nodes, used for examples in Opsahl, Agneessens, Skvoretz (2010) Social Networks 32(2010):245-251

### Usage

OpsahlEtAl10

### Format

A numeric matrix with 5 rows and 5 columns

### Source

<DOI:10.1016/j.socnet.2010.03.006>
opt_gpv

**Optimize Generalized Path Value**

**Description**
Identify the path of optimal generalized path value from a source node to a target node.

**Usage**
opt_gpv(sociomatrix, source, target, p = Inf, node_costs = NULL)

**Arguments**
- `sociomatrix`: a nonnegative, real valued sociomatrix.
- `source`: an integer index corresponding to a node in sociomatrix.
- `target`: an integer index corresponding to a node in sociomatrix.
- `p`: a nonnegative real number that sets the 'p-norm' parameter for generalized path value calculation.
- `node_costs`: a list of costs, in order, of all nodes represented in the sociomatrix, all are assumed 0 if unspecified.

**See Also**
gpv to calculate the value of a user-specified path, all_opt_gpv to simultaneously identify the optimal paths from any source node to any target node.

opt_ppv

**Optimize Probabilistic Path Value**

**Description**
Identify the path of optimal probabilistic path value from a source node to a target node.

**Usage**
opt_ppv(sociomatrix, source, target, odds_scale = 1, odds_scale_by_node = NULL)

**Arguments**
- `sociomatrix`: a nonnegative, real valued sociomatrix.
- `source`: an integer index corresponding to a node in sociomatrix.
- `target`: an integer index corresponding to a node in sociomatrix.
- `odds_scale`: a nonnegative real number indicating the observed tie strength value that corresponds to 1-1 transmission odds.
- `odds_scale_by_node`: sets a transfer odds scale for each node in a probabilistic path value calculation.
peay_average_path_value

**Yang and Knoke’s Average Path Value**

**Description**

Calculates 'APV' (Average Path Value) as defined in Yang, Knoke (2001) Called peay_average_path_value in homage to E.R. Peay, who defined path length in 1980.

**Usage**

peay_average_path_value(sociomatrix, path)

**Arguments**

- **sociomatrix** a nonnegative, real valued sociomatrix.
- **path** an integer vector of node indices from sociomatrix.

**See Also**

peay_path_value

**Examples**

```r
## Calculate 'APV' of a path in a sociomatrix
peay_average_path_value(YangKnoke01, path = c(1,2,5))

## This path doesn't exist
peay_average_path_value(YangKnoke01, path = c(1,2,4,5))
```

---

peay_path_value

**Peay’s Path Value Measure**

**Description**

Calculates path value as defined in Peay (1980). That is, returns the value of the weakest connection in the path, if all edges exist. Otherwise, returns 0.

**Usage**

peay_path_value(sociomatrix, path)

**Examples**

```r
## Calculate 'APV' of a path in a sociomatrix
peay_average_path_value(YangKnoke01, path = c(1,2,5))

## This path doesn't exist
peay_average_path_value(YangKnoke01, path = c(1,2,4,5))
```
Arguments

sociomatrix  a nonnegative, real valued sociomatrix.
path        an integer vector of node indices from sociomatrix.

See Also

peay_average_path_value

Examples

```r
## Calculate Peay's Path Value along a path in a sociomatrix
peay_path_value(YangKnoke01, path = c(1,2,5))

## This path doesn't exist
peay_path_value(YangKnoke01, path = c(1,2,4,5))
```

Description

Given a real valued sociomatrix, a path, and an optional odds_scale, ppv calculates the transmission odds for the path and returns the transmission odds times odds_scale so the result can be directly compared with observed tie strengths.

Usage

```r
ppv(sociomatrix, path, odds_scale = 1, odds_scale_by_node = NULL)
```

Arguments

sociomatrix  a nonnegative, real valued sociomatrix.
path        an integer vector of node indices from sociomatrix.
odds_scale  a nonnegative real number indicating the observed tie strength value that corresponds to 1-1 transmission odds
odds_scale_by_node sets a transfer odds scale for each node in a probabilistic path value calculation.

Details

We assume that observed tie strengths in sociomatrix are linearly proportional to transmission odds. That is, if the transmission odds for a strength 1 tie are 1 to 1, the transmission odds for a strength 5 tie are 1 to 5.
unpack

See Also

opt_ppv to identify the path of optimal 'ppv' between two nodes and all_opt_ppv to identify the optimal paths between all pairs of nodes. Calling generate_proximities with mode = 'ppv' returns a matrix 'ppv' values for the optimal paths between all pairs of nodes.

Examples

```r
## Calculate ppv along a path in a sociomatrix
ppv(YangKnoke01, path = c(1,2,5), odds_scale = 3)

## This path doesn't exist
gpv(YangKnoke01, path = c(1,2,4,5))
```

Description

Unpacks a Path from a Dijkstra-Format Spanning Tree

Usage

```r
unpack(tree, source, target)
```

Arguments

- `tree`: a Dijkstra-format tree returned by all_opt_gpv or all_opt_ppv
- `source`: an integer index corresponding to a node in sociomatrix
- `target`: an integer index corresponding to a node in sociomatrix

Details

Returns NA if a path does not exist
<table>
<thead>
<tr>
<th>YangKnoke01</th>
<th>Example data from Yang, Knoke (2001)</th>
</tr>
</thead>
</table>

**Description**


**Usage**

YangKnoke01

**Format**

a numeric matrix with 5 rows and 5 columns

**Source**

<DOI: 10.1016/S0378-8733(01)00043-0>
Index

* datasets
  BuchDarrah19, 4
  OpsahlEtAl10, 9
  YangKnoke01, 14

all_opt_gpv, 2, 8, 10
all_opt_ppv, 3, 11, 13

binary_distance, 3
BuchDarrah19, 4

dijkstra_inf, 4
dijkstra_nodes, 5

flament_average_path_length, 5, 6
flament_path_length, 6, 6

generate_proximities, 7, 8, 13
gpv, 3, 7, 8, 10
gretel, 9
gretel-package (gretel), 9

OpsahlEtAl10, 9
opt_gpv, 3, 8, 10
opt_ppv, 3, 10, 13

peay_average_path_value, 11, 12
peay_path_value, 11, 11
ppv, 3, 7, 11, 12

unpack, 13

YangKnoke01, 14